Architecture design decision maps for software sustainability

Lago, Patricia

published in
2019 IEEE/ACM 41st International Conference on Software Engineering
2019

DOI (link to publisher)
10.1109/ICSE-SEIS.2019.00015

document version
Publisher's PDF, also known as Version of record

Cite this work as:
Lago, P. (Accepted/In press). Architecture Design Decision Maps for Software Sustainability. Paper presented at IEEE/ACM International Conference on Software Engineering, Montreal, Canada. 2019. https://goo.gl/z651FB

Architecture Design Decision Maps for Software Sustainability

Patricia Lago
Vrije Universiteit Amsterdam, The Netherlands
p.lago@vu.nl

***Abstract*—In software engineering, sustainability can be defined as the "capacity to endure" and to "preserve the function of a system over an extended period of time". These definitions mainly point towards technical sustainability over time. Sustainability, however, may entail a much broader scope including economic, social and environmental sustainability as well.**

In spite of the exciting hype around sustainability, we are very much lacking suitable instruments to design software-intensive systems that are sustainable and enable sustainability goals. To fill this gap, we advocate the treatment of sustainability as a software quality property and define a software sustainability assessment method that helps make sustainability-driven design decisions. The method essentially relies on the definition of so-called *decision maps*, i.e. views aimed at framing the architecture design concerns around the four sustainability dimensions mentioned above - technical, economic, social and environmental sustainability. This paper presents the notion of decision map. We use two illustrative examples extracted from industrial projects, to summarize our lessons learned and reflections.

***Index Terms*—Software architecture; Sustainability; Architecture design decisions; Architecture assessment; Decision map;**

I. Introduction

In software engineering, sustainability is often defined as the "capacity to endure" (borrowed from [19]) and to "preserve the function of a system over an extended period of time" [14]. These definitions mainly point towards what we call *technical sustainability*, i.e. "the preservation of the long-term use of software-intensive systems and their appropriate evolution in an execution environment that continuously changes" [12].

Thanks to its growing popularity, more and more scientific works in software engineering and software architecture address, or at least mention, sustainability from a technical perspective e.g. [3]. It must be observed, however, that many such works often share two weaknesses: (i) they confuse the notion of impact (which is measured in a certain point in time) with the notion of sustainability (which is a phenomenon observable only over a significant period of time); (i) they limit the notion of sustainability to technical aspects (e.g. evolvability, maintainability, erosion) and sometimes environmental ones (e.g. energy consumption, power efficiency). Sustainability, however, entails a much broader scope including economic and social aspects as well.

II. The Vision

In software architecture we are mostly used to think in terms of technical impact [3] and economic concerns [17]. Only recently we have started thinking in terms of environmental ones (e.g. [2], [21]). In one way or another, we are also more used to compartmentalize these concerns, whereas sustainability is a matter of assessing the big picture. To this end, we argue that tradeoff analysis provides the perfect mechanism to simultaneously consider all sustainability aspects that might be relevant for a certain system, and hence lay the foundation for a much better understanding of what is sustainability in software-intensive systems. We also argue that software architecture is an ideal abstraction level to gather the above-mentioned *big picture*: by combining architecture assessment methods with sustainability tradeoffs, we would be finally able to deliver software that will support sustainability *by design*.

In spite of the exciting hype around sustainability, we are very much lacking suitable instruments to design software-intensive systems that are truly sustainable or that enable sustainability goals. To fill this gap, we advocate the treatment of sustainability as a software quality property and defined a software sustainability assessment method, called SoSA [12], that helps making sustainability-driven design decisions. The method essentially relies on the definition of so-called "decision maps" framing the architecture design concerns around the four sustainability dimensions mentioned above - technical, economic, social and environmental sustainability.

In this paper, we present the notion of decision map resulting from over four years of research in collaboration with the private- and public sectors. We use two illustrative examples extracted from a selection of our industrial projects to summarize our lessons learned and reflections.

Below we present the decision maps and lessons learned (Sect. III), related works (Sect. IV) and conclusions (Sect. V).

III. Decision Maps

Architecture evaluations should, among other, *"support decision making where architectures are involved"*, *"assess the quality of architectures with respect to their intended purpose"* and *"determine whether architecture entities address their intended purpose"* [11]. If one architecture's intended purpose is sustainability, we should provide architects suitable instruments to make decisions that lead to some stated sustainability purposes, or concerns. This is exactly the aim of our decision maps, i.e. making explicit the sustainability concerns that should be considered by an architecture [11].

The notion of decision map (DM) is the result of multiple research projects (e.g. [7]) carried out incrementally in collab-

oration with industry and the public sector. After introducing the visual notation, the following uses the example DMs (see Fig. 2) resulting from the projects summarized in Table I to draw some of our main lessons learned.

A. The Notation

The DM notation (exemplified in Fig. 1) essentially frames the expected impact of a software architecture on the target sustainability concerns. Accordingly, the notation entails sustainability impacts (see in the Figure, the areas in different shades of grey), sustainability architecture design concerns (the colored boxes), and the effects among concerns (the arrows between boxes).

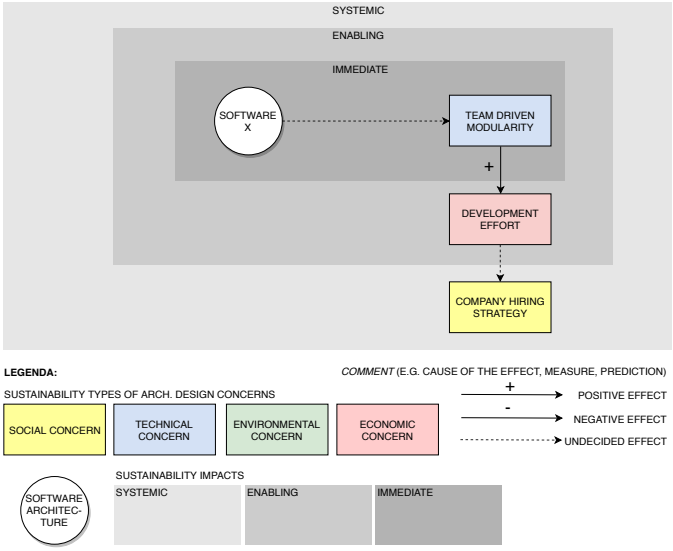


Fig. 1: DM Notation

1) On Sustainability Impacts: Our definition of sustainability impact builds upon that of Hilty et al. [9] along three levels: **Immediate impacts** refer to changes that are immediately observable. These are the concerns that are addressed within the current software project, and are expected to be directly addressed by the architecture entities. In the example of Fig. 1, the DM models that the architecture modularity is expected to reflect the structure of the development team. **Enabling impacts** arise from use over time. This includes the opportunity to consume more (or less) resources, but also shorten their useful life by obsolescence (when we buy a new smart phone just because incompatible with newer applications) or substitution (when e-book readers replace printed books). In our example, team-driven modularity is expected to enable a positive impact on the development effort. While not directly measurable within the scope of the current software project, this concern can be monitored over multiple projects, hence provide understanding on the extent of such positive impact. **Systemic impacts**, in turn, refer to persistent changes observable at the macro level. This includes behavioral change and economic structural change. This may translate into (negative) rebound effects by converting efficiency improvements into additional consumption, or new risks - like our dependence on ICT networks that makes a digital society also vulnerable. In our example, a sustained reduction of the development effort required by software projects may result in less developers needed in the first place, or employees with different technical competences, and hence a change in the company hiring strategy.

2) On Sustainability Concerns: Architecture design concerns can be of four sustainability types [12], [13]: **Technical sustainability** addresses the long-term use of software-intensive systems and their appropriate evolution in an execution environment that continuously changes. **Economic sustainability** focuses on preserving capital and (economic) value. **Social sustainability** focuses on supporting current and future generations to have the same or greater access to social resources by pursuing generational equity. For software-intensive systems, this dimension encompasses the direct support of social communities in any domain, as well as the support of activities or processes that indirectly create benefits for social communities. **Environmental sustainability** aims at improving human welfare while protecting natural resources. For software-intensive systems, this dimension aims at addressing ecologic concerns, including energy efficiency and ecologic awareness creation.

In our example, team driven modularity expresses a technical concern, development effort translates into economic impacts, and company hiring strategy reflects the organizational social structure and as such is a concern of social nature.

3) On the Effects: We have identified three types of effects among software architecture entities and concerns: positive, negative, and undecided. Their semantics is quite self-explanatory, and this is in our experience an essential requirement to keep the decision maps simple and intuitive enough as an instrument for decision makers with different backgrounds and competencies.

Decision maps are also meant to be used and re-used across various projects. As such, incremental learning will allow effects to evolve and consolidate over time. For example, by observing the impact of multiple projects on the concerns shown in Fig. 1, the *undecided* impact on the company hiring strategy might become positive (e.g. if the company is agile enough to adapt timely) or negative (e.g. if the company decides for a rapid turnover strategy, with a possible negative effect on the working atmosphere). Either ways, the DM can be used to capture the decisions and the consequent effects. If so, it can provide a valid yet simple decision making instrument.

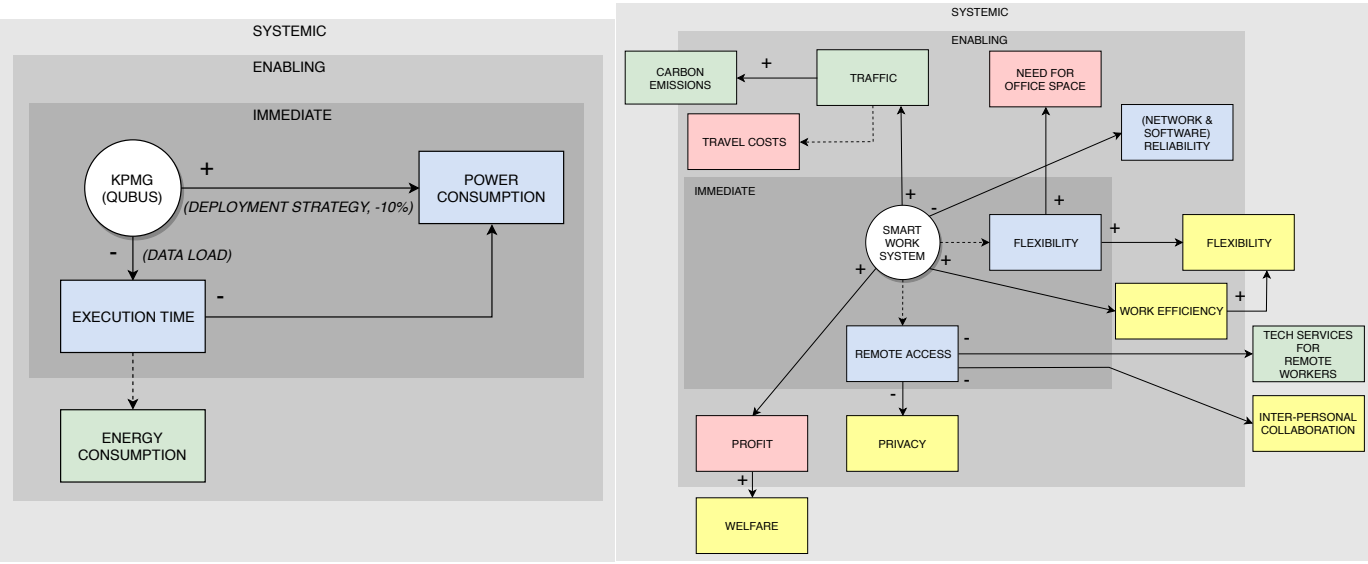
B. Lessons Learned from Example Projects

Among the projects we carried out over the years, we picked two to illustrate how DMs can help the analysis of the sustainability concerns. For each example, we summarize our reflection and our insights in the form of lessons learned. The projects are summarized in Table I.

In the **KPMG Qubus platform** (Fig. 2a) the target concerns were about the effect of different software releases on energy consumption while maintaining a satisfactory level

Project Title	Abstract	Concerns	DM
KPMG Qubus [20]	The object of this study is a platform named Qubus, which supports Governance, Risk, and Compliance (GRC) processes and is developed and maintained by KPMG. The selected software product supports a wide range of enterprise management processes including financial control management, strategic and operational risk management, compliance management and external audit management.	What is the impact of different software releases on energy consumption?	Fig. 2a
Smart Work [1]	In 2025 urban transport in the Amsterdam Metropolitan Area will be emission free. To get there, the next few years will witness major disruptions in transportation that will hinder e.g. reaching the office and school in time. The solution is to seek alternative ways to work and study so that to remove the need to physically reach one work- or study places. In doing so, however, the way we work, study, interact with out family, spend our free time will change forever. How do we make sure that this transformation toward a smart and sustainably city and lifestyle will be smooth and successful? And how do we foresee the behavioural changes that will mark our futures?	How to choose between "travel" and "untravel" (i.e. substitute activities that require physical mobility with others that allow virtual presence)? What are the long-term implications of virtual mobility? For instance, when workers do need using physical offices much less, or the boundary between work and private will be blurred?	Fig. 2b

TABLE I: Summary of the Projects



(a) KPMG Qubus

(b) Smart Work Service

Fig. 2: Decision Maps. [See notation in Fig. 1]

of performance (expressed in terms of execution time). The initial DM hypothesized a positive effect of successive releases on POWER CONSUMPTION, and an hypothetical corresponding negative effect on EXECUTION TIME. Both concerns are clearly technical. Actual measures, however, showed that: (i) the main design decision responsible for variations in power consumption regards the *deployment strategy* with an effect measured up to 10%; (ii) the main feature responsible for negative effects on execution time is *data load*, i.e. the exchange of data between client- and server side necessary for executing the user services; in turn, execution time has a negative effect on power consumption; (iii) surprisingly, we could not determine any significant correlation between execution time and ENERGY CONSUMPTION, leaving the effect of the first on the second as *undecided*.

Lesson #1: Decision Maps can be refined with the actual causes of effects and effect measures.

- Initially, DMs can be used for framing the **foreseen effects** of early architecture design decisions on the target concerns.
- After architecture evaluation or implementation, observations and measures can be used to refine the DMs with the **actual effects** on the target concerns.

The exploratory nature of the **Smart Work** project makes the decision maps especially useful for identifying concerns of various sustainability types and regarding different impact levels (Fig. 2b). By drawing an early decision map, the architect can reason about the potential pros and cons of alternative solutions, and explore the design space before making specific design decisions. Accordingly, the focus of such an early DM is on mapping the sustainability concerns and their potential crosscutting effects, before making design decisions.

This project was carried out in 2016, in collaboration with the Amsterdam Smart City organisation, and was followed by a focus group in the 28th International Conference on Advanced Information Systems Engineering (CAiSE). The decision map in Fig. 2b captures the refined interpretation chosen as the preferred DM for the problem of providing software services for flexible work. We observe that: (i) like in the previous project, also here concerns with immediate impact are technical; (ii) given the broad scope of the problem, relevant concerns cover all sustainability types; (iii) in some cases, the same concern (see FLEXIBILITY in the Figure) can address multiple sustainability aspects (e.g. *technical* flexibil-

ity is necessary to provide a satisfactory quality of service for home-working, and if present it can have a positive impact on *social* flexibility like the ability to balance work-, family- and leisure time).

Lesson #2: Decision Maps can be used for design space exploration, with a major focus on characterizing and scoping sustainability concerns.

- DMs can effectively illustrate the problem space in terms of the relevant sustainability concerns and their potential cross-cutting effects.
- A concern can possibly capture multiple sustainability perspectives (cf. flexibility being both technical and social).
- The three levels of impact provide the natural context to reason about the implications of decisions affecting concerns from the short- (immediate and enabling impact) to the longer term (systemic impact).

In addition to the lessons learned, we drew two general observations. ◆ Thanks to its simplicity, DMs can uncover interesting phenomena that help informed design decision making further. E.g., the effects between socio-technical flexibility and work efficiency in Fig. 2b reveal a so-called network effect [16] where work efficiency enabled by smart work services has the potential to further enforce social flexibility. This, in turn, can have an important systemic impact on modern society (and for this reason, in the DM the associated concern is placed across the two dimensions). By uncovering this network effect, architects can design for achieving this potential impact, and put in place measures to observe if (when implemented) the architecture does realize the designed-for sustainability goals. ◆ DMs have been conceived for designing software architectures addressing sustainability concerns. However, they are not "software-specific" and, as regularly observed by our industrial partners, they can be extended to the notion of enterprise architecture [8] and used to uncover the potential networks of stakeholders linked to the mapped sustainability concerns. In doing so, they can facilitate the creation of innovative sustainability business models.

IV. Related Work

This work stems from our early interpretation of sustainability as a software quality property [13]. From there, we take an architecture design perspective, while Becker et al. [4] take a requirements engineering one. Most researches in the field focus on specific sustainability aspects, e.g., Cai et al. [5] use design rules to detect software architecture flows over time [technical sustainability]; Hindle [10] relates software change and energy consumption, and Li et al. [15] present practices that help reduce the energy consumption of mobile apps [environmental sustainability]; Widdicks et al. [22] study socio-technical sustainability of individuals, while Tamburri et al. [18] do so for development teams and whole organizations [social sustainability]. Our work is orthogonal as DMs frame any sustainability concerns.

V. Conclusion and Future Work

This paper presents and discusses the notion of *decision maps* framing the sustainability-related architecture design concerns. As future work, we are planning to follow-up our lessons learned. E.g., regarding Lesson #1, we are extending the DM notation to link concerns and metrics, then used on implemented architectures to gather measures and extend our sustainability-quality model [6].

Acknowledgment

Thanks go to Amsterdam Smart City and KPMG to borrow their projects for this research, the many students of the VU Amsterdam who helped inspiring this research, and Wouter-Paul Trienekens (CGI Group) for his valuable feedback.

References

[1] Amsterdam smart city. https://amsterdamsmartcity.com.

[2] F. Alizadeh Moghaddam, G. Procaccianti, G. A. Lewis, and P. Lago. Empirical validation of cyber-foraging architectural tactics for surrogate provisioning. *The Journal of systems and software*, 138:37–51, 2018.

[3] P. Avgeriou, M. Stal, and R. Hilliard. Architecture sustainability [guest editors' introduction]. *Software, IEEE*, 30(6):40–44, 2013.

[4] C. Becker, S. Betz, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. Venters. Requirements: The key to sustainability. *IEEE Software*, 33(1):56–65, Jan. 2016.

[5] Y. Cai, L. Xiao, R. Kazman, R. Mo, and Q. Feng. Design rule spaces: A new model for representing and analyzing software architecture. *IEEE Transactions on Software Engineering*, (1):1–1.

[6] N. Condori-Fernandez and P. Lago. Characterizing the contribution of quality requirements to software sustainability. *Journal of Systems and Software*, 137:289–305, 3 2018.

[7] S. Espana and P. Lago. Software sustainability assessment (SoSA) exercise report. Technical report, Vrije Universiteit Amsterdam, 2016.

[8] D. Greefhorst and E. Proper. *Architecture Principles: The Cornerstones of Enterprise Architecture*. Springer, Apr. 2011.

[9] L. M. Hilty et al. The relevance of information and communication technologies for environmental sustainability – a prospective simulation study. *Environmental Modelling & Software*, 21(11):1618–1629, 2006.

[10] A. Hindle. Green mining: A methodology of relating software change and configuration to power consumption. *Empirical Softw. Engg.*, 20(2):374–409, Apr. 2015.

[11] ISO/IEC TC /SC 7/WG 42. ISO/IEC/IEEE 42030:2017 - Enterprise, systems and software – Architecture evaluation framework. Technical report, International Organization for Standardization (ISO), Oct. 2017.

[12] P. Lago. SoSA: A Software Sustainability Assessment method. http://goo.gl/HuY6tf, Jan. 2016.

[13] P. Lago et al. Framing sustainability as a property of software quality. *CACM*, 58(10):70–78, 2015.

[14] P. Lago and B. Penzenstadler. Editorial: Reality check for software engineering for sustainability—pragmatism required. *Journal of Software: Evolution and Process*, 29(2), Feb. 2017.

[15] D. Li and W. G. J. Halfond. An investigation into energy-saving programming practices for android smartphone app development. In *Proceedings of the 3rd International Workshop on Green and Sustainable Software (GREENS)*. ACM, 2014.

[16] G. G. Parker, M. W. Van Alstyne, and S. P. Choudary. *Platform Revolution: How Networked Markets Are Transforming the Economy and How to Make Them Work for You*. 2016.

[17] E. R. Poort and H. van Vliet. RCDA: Architecting as a risk- and cost management discipline. *JSS*, 85(9):1995–2013, 2012.

[18] D. A. Tamburri, P. Kruchten, P. Lago, and H. v. Vliet. Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications*, 6(1):10, May 2015.

[19] UN World Commission on Environment and Development. Report: Our Common Future. In *UN Conf. on Environment and Development*, 1987.

[20] R. Verdecchia, G. Procaccianti, I. Malavolta, P. Lago, and J. Koedijk. Estimating energy impact of software releases and deployment strategies: The KPMG case study. In *ESEM*, pages 257–266, Nov. 2017.

[21] R. Verdecchia, R. A. Saez, G. Procaccianti, and P. Lago. Empirical evaluation of the energy impact of refactoring code smells. In *International Conference on ICT for Sustainability*, ICT4S, 2018.

[22] K. Widdicks, T. Ringenson, D. Pargman, V. Kuppusamy, and P. Lago. Undesigning the Internet: An exploratory study of reducing everyday internet connectivity. In *ICT4S*, pages 384–397, 2018.